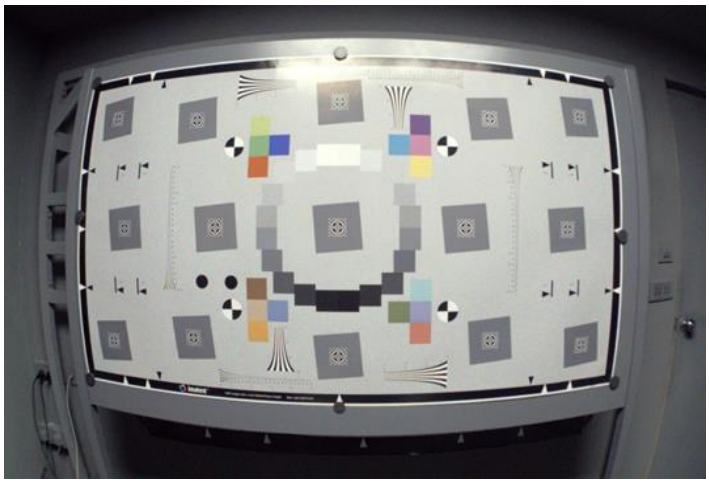
PROBLEM STATEMENT :

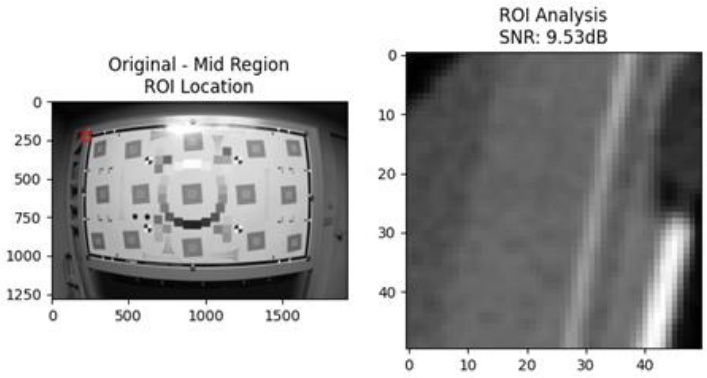
Assignment -2: Denoise and Sharpness Techniques

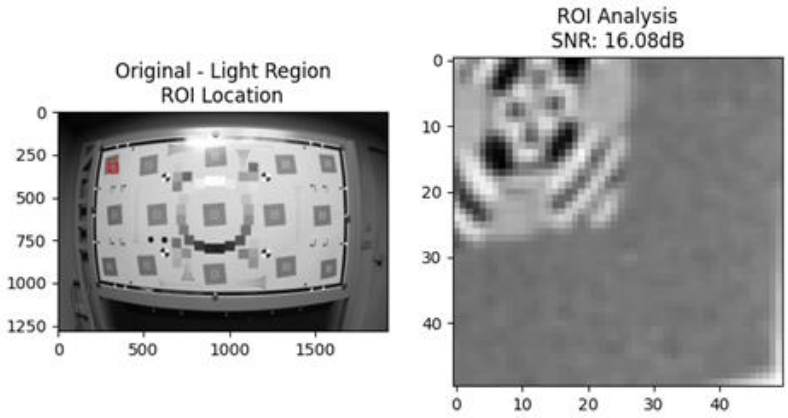
- Objective – Implement different techniques of denoise and sharpness and assess **image quality**
- Task –
  - Denoise
    - Implement median and bilateral filter. Compare with gaussian filter implemented in Assignment 1
    - Implement **AI model** to denoise image and compare with above traditional methods
    - Compute spatial Signal to noise ratio for 3 different gray tones as shown **(in next slide)** for each of the methods implemented. Refer https://www.imatest.com/imaging/noise/
  - Edge enhancement
    - Implement Laplacian filter based enhancement and compared with the method implemented in Assignment 2.
    - Compute edge strength based on gradient based approach for each of the methods implemented
  - Use ISP pipeline from Assignment 1 for processing the input raw image
- Input – 12bit Bayer Raw image. Output – RGB channel with 24 bits per pixel (8 bits for each channels).
- Tools to view RAW and output image (select appropriate formats).
  - PixelViewer
  - Irfanview with RAW plugin
  - Configuration to be used for input – Bayer – 12bits, GRBG, 1920x1280
  - AI model - Tensorflow for denoise CNN model. Optional to use training process else use pre-trained models such as U-net, FFDNet etc.
- Generate report with all the comparison and image quality metrics computed

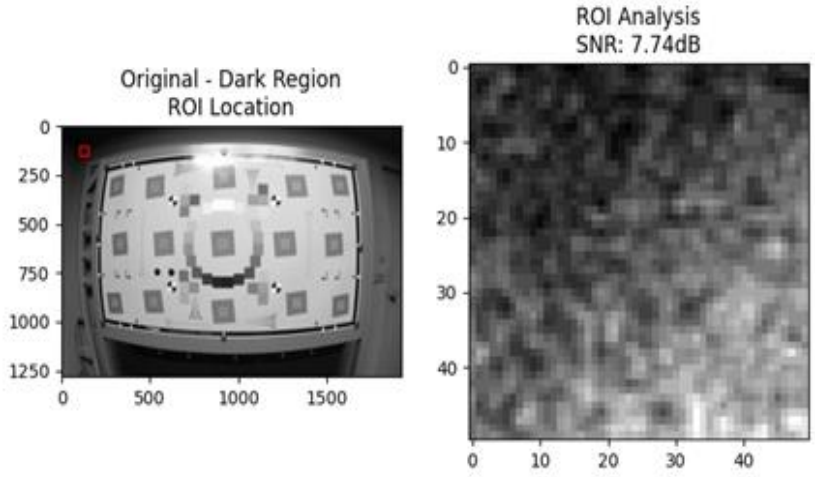
Github Link: https://github.com/yashbudhia/emmetra

Input image	AI denoised image
	
	

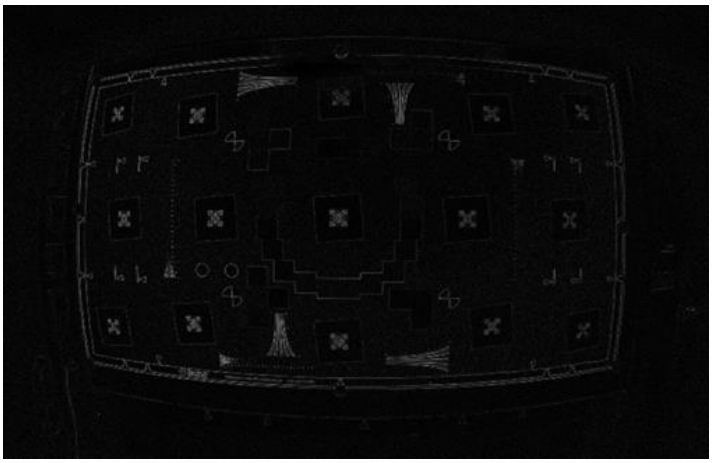
ORIGINAL IMAGE :



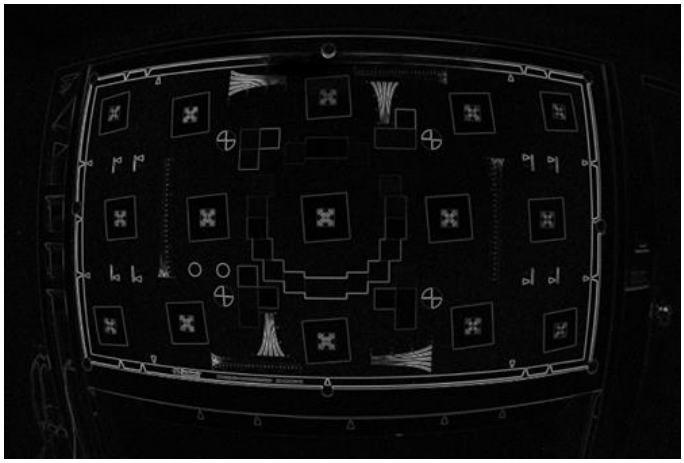




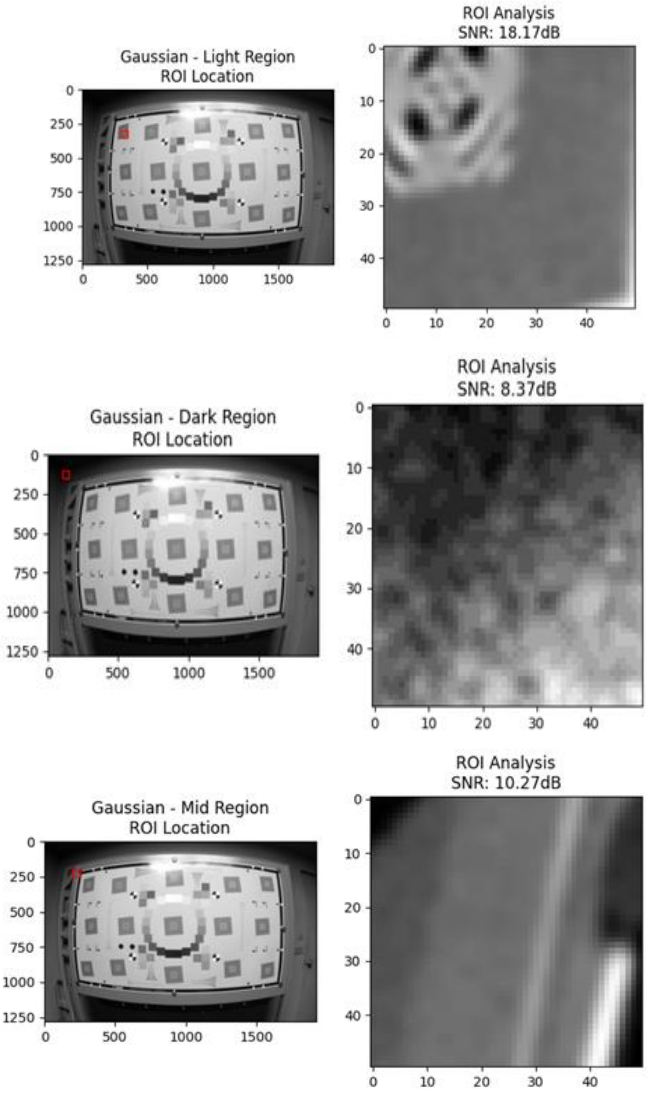
Original Laplacian Edges :



Original Sobel edges :

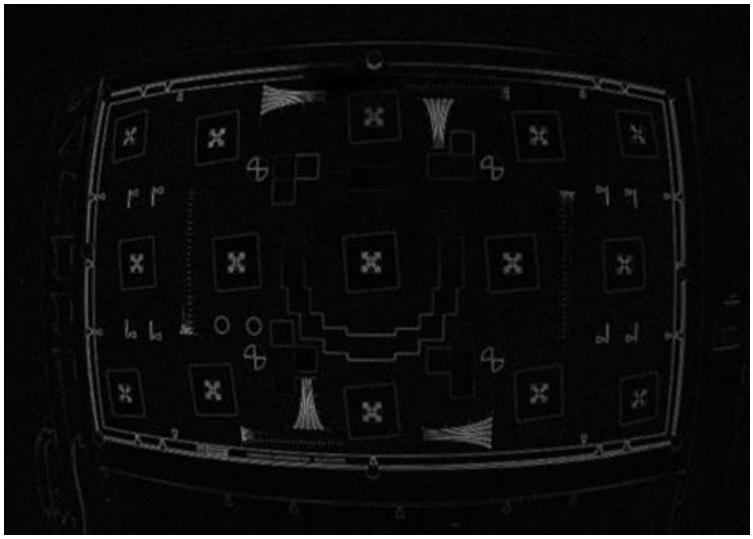


With GAUSSIAN Filter

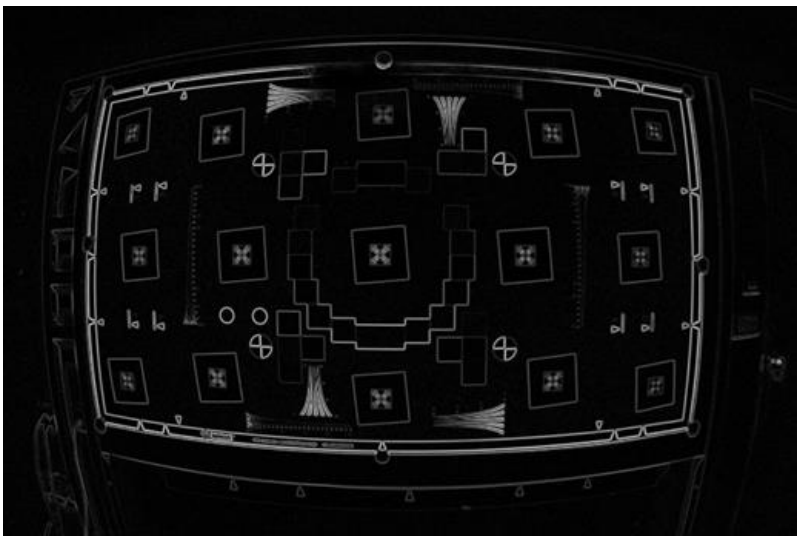


:

Gaussian Laplacian edges :



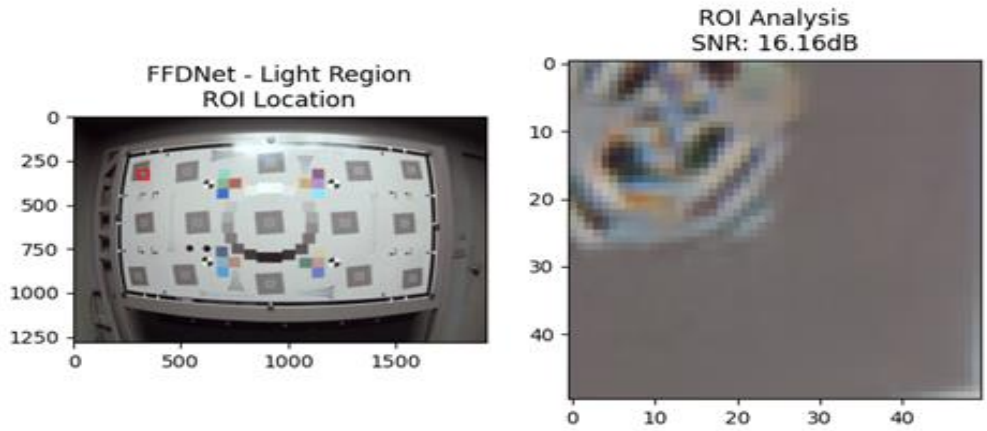
Gaussian Sobel edges

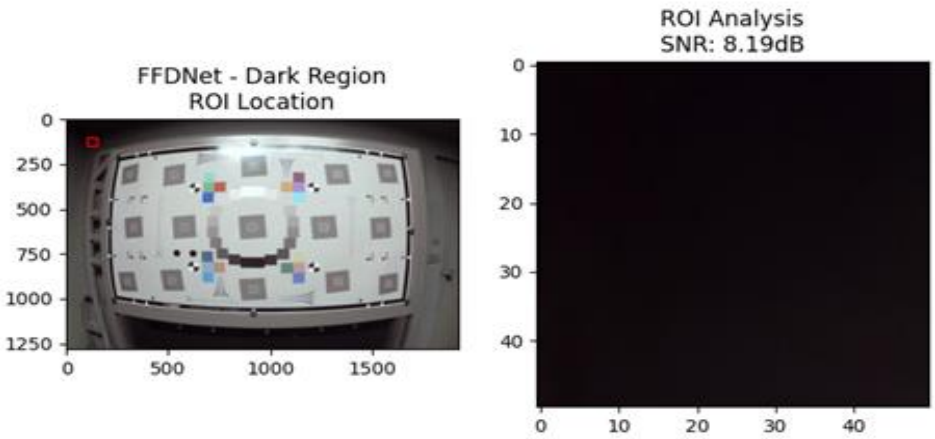


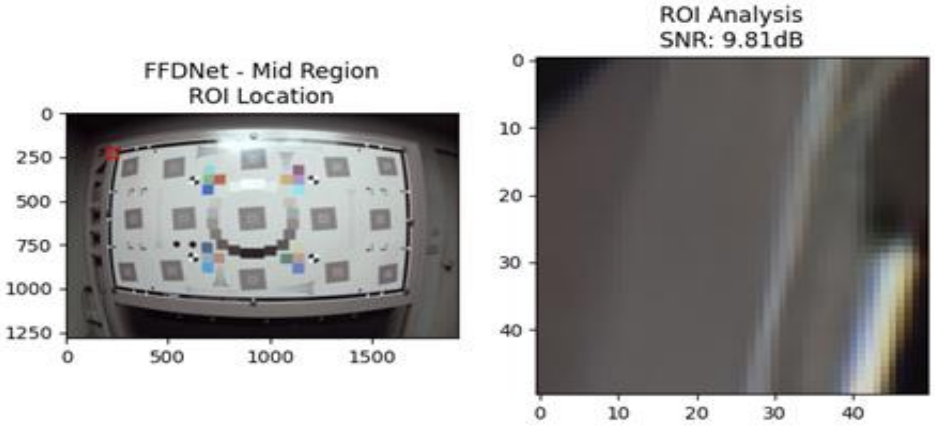
Gaussian:

- ★ Gaussian filtering is a widely used image processing technique that applies a Gaussian kernel to smooth and reduce noise in an image.
- ★ The Gaussian kernel is a bell-shaped function that assigns higher weights to the pixels closer to the centre and lower weights to the pixels farther away.
- ★ Gaussian filtering is effective in removing high-frequency noise while preserving the overall structure and edges of the image.

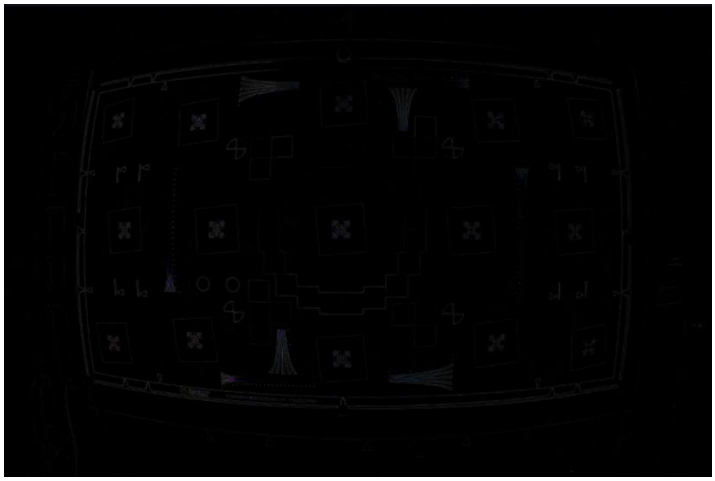
Using FFDNET Model for Denoising



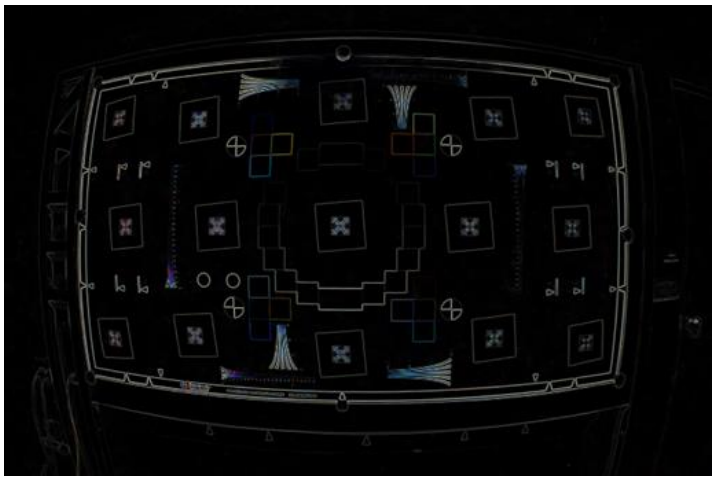




FFDnet Laplacian edges :



FFDnet Sobel edges :

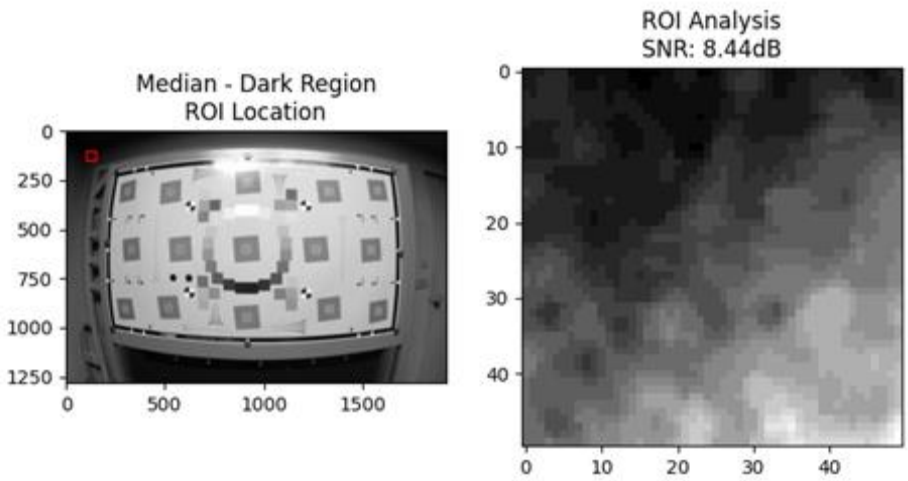


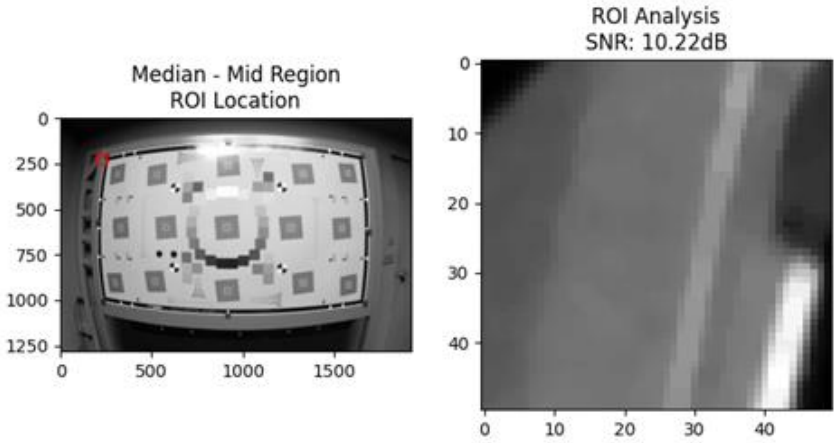
FFDNet :

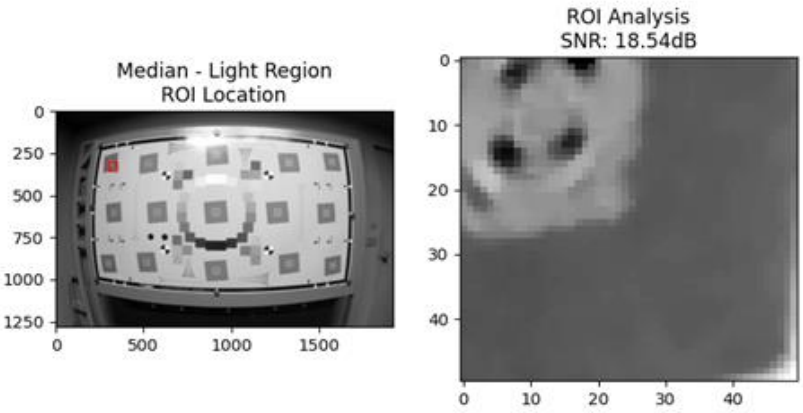
- ★ FFDNet (Flexible Directional Denoising Network) is a deep learning-based image denoising model that uses a convolutional neural network architecture.
- ★ The FFDNet model is designed to be flexible and adaptable, allowing it to handle various levels of noise in the input image.

By incorporating residual blocks and attention mechanisms, the FFDNet model aims to effectively remove noise while preserving important image details and structures.

MEDIAN Filtering

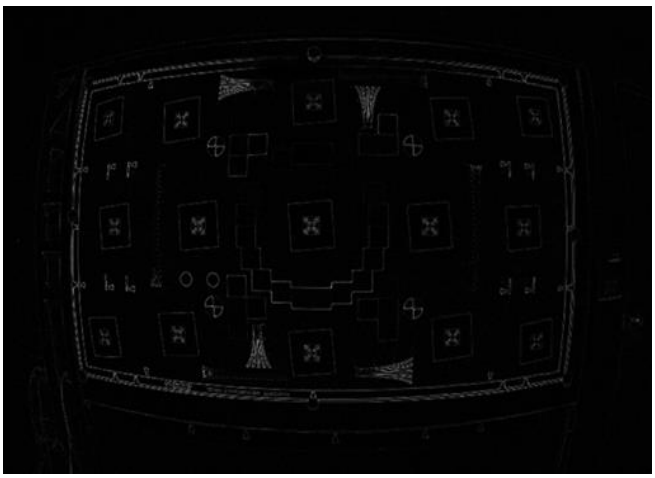


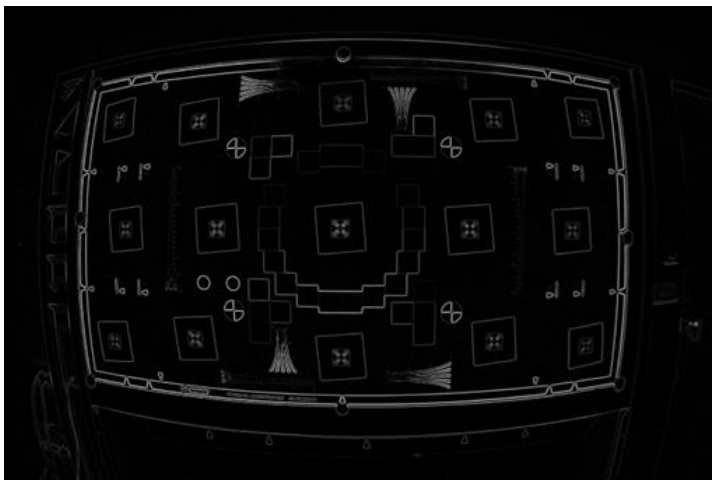




Median Laplacian Edges :

Median Sobel edges :



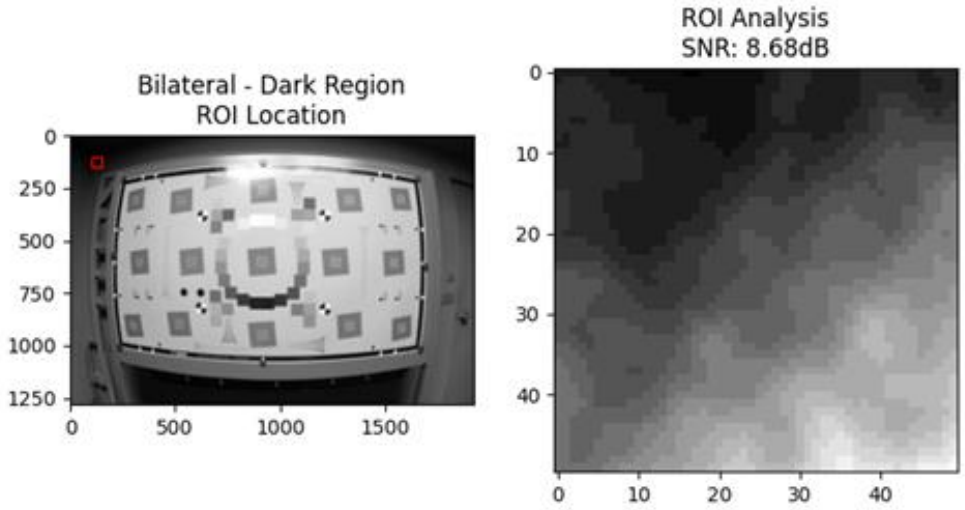


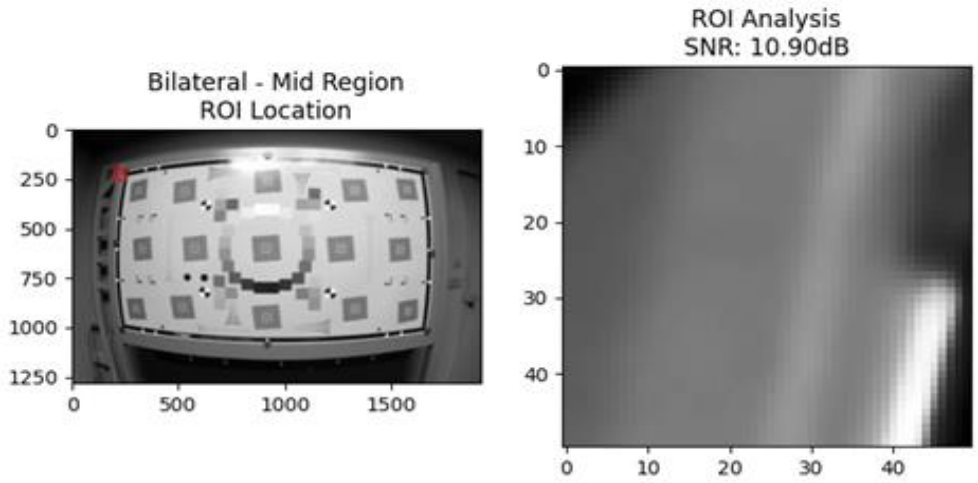
Median:

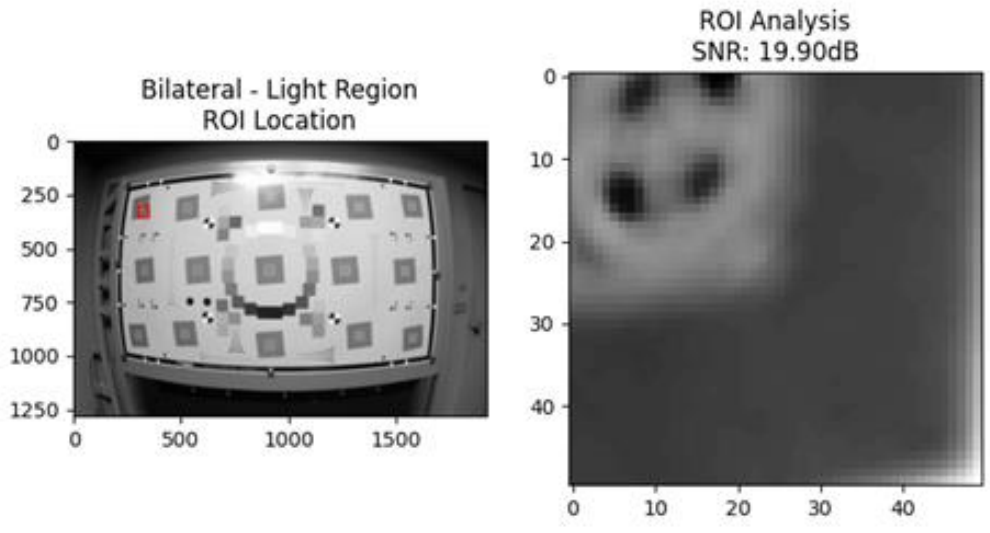
- ★ Median filtering is a non-linear image processing technique that replaces each pixel with the median value of its neighbouring pixels.
- ★ The median filter is effective in removing impulsive noise, such as salt-and-pepper noise, while preserving edges and other important image features.

★ Median filtering is often used as a preprocessing step before applying other image processing algorithms, as it can help reduce noise without significantly distorting the underlying image structure.

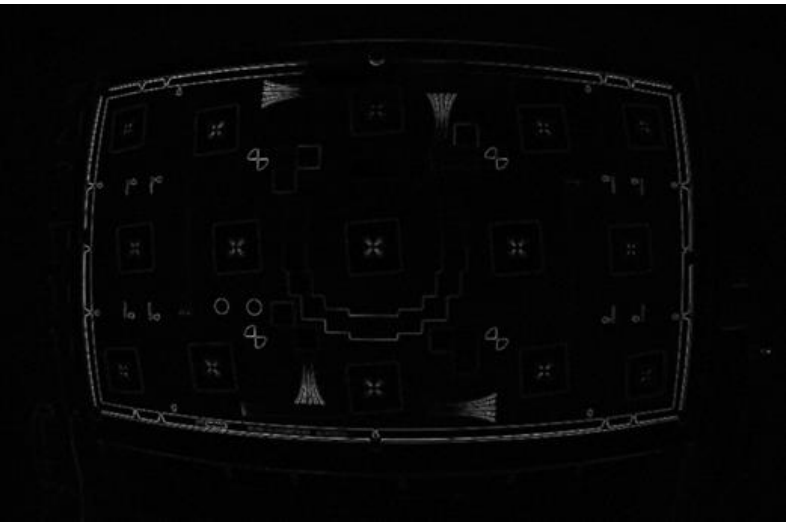
BILATERAL FILTERING



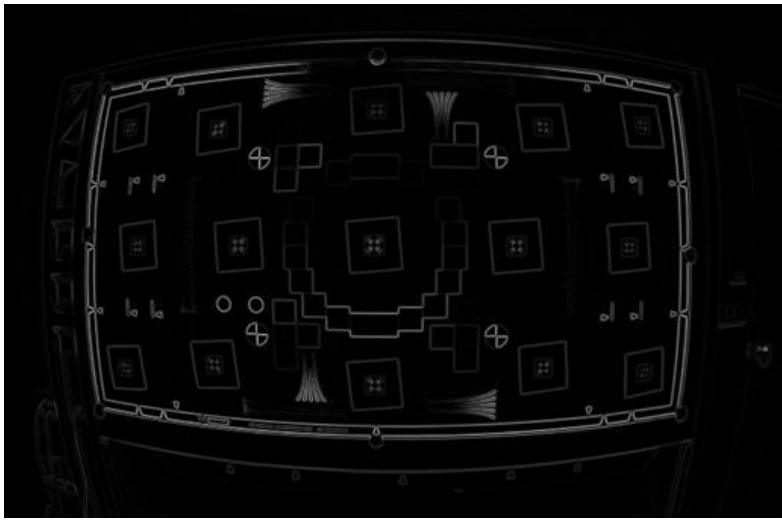




Bilateral Laplacian :



Bilateral Sobel Edges :



Bilateral:

- ★ Bilateral filtering is an advanced image smoothing technique that combines the concepts of Gaussian filtering and edge-preserving filtering.
- ★ It applies a Gaussian filter in the spatial domain, but also considers the similarity of pixel intensities, allowing it to preserve sharp edges while still reducing noise.
- ★ Bilateral filtering is particularly useful for removing noise while maintaining important image details and structures.

Laplacian:

- ★ The Laplacian operator is a second-order derivative used in image processing to detect edges and enhance image sharpness.
- ★ It is often used in combination with other filters, such as Gaussian or Sobel, to highlight the edges and boundaries within an image.
- ★ The Laplacian-of-Gaussian (LoG) is a common technique that combines Gaussian smoothing and Laplacian edge detection to enhance the visibility of edges in an image.

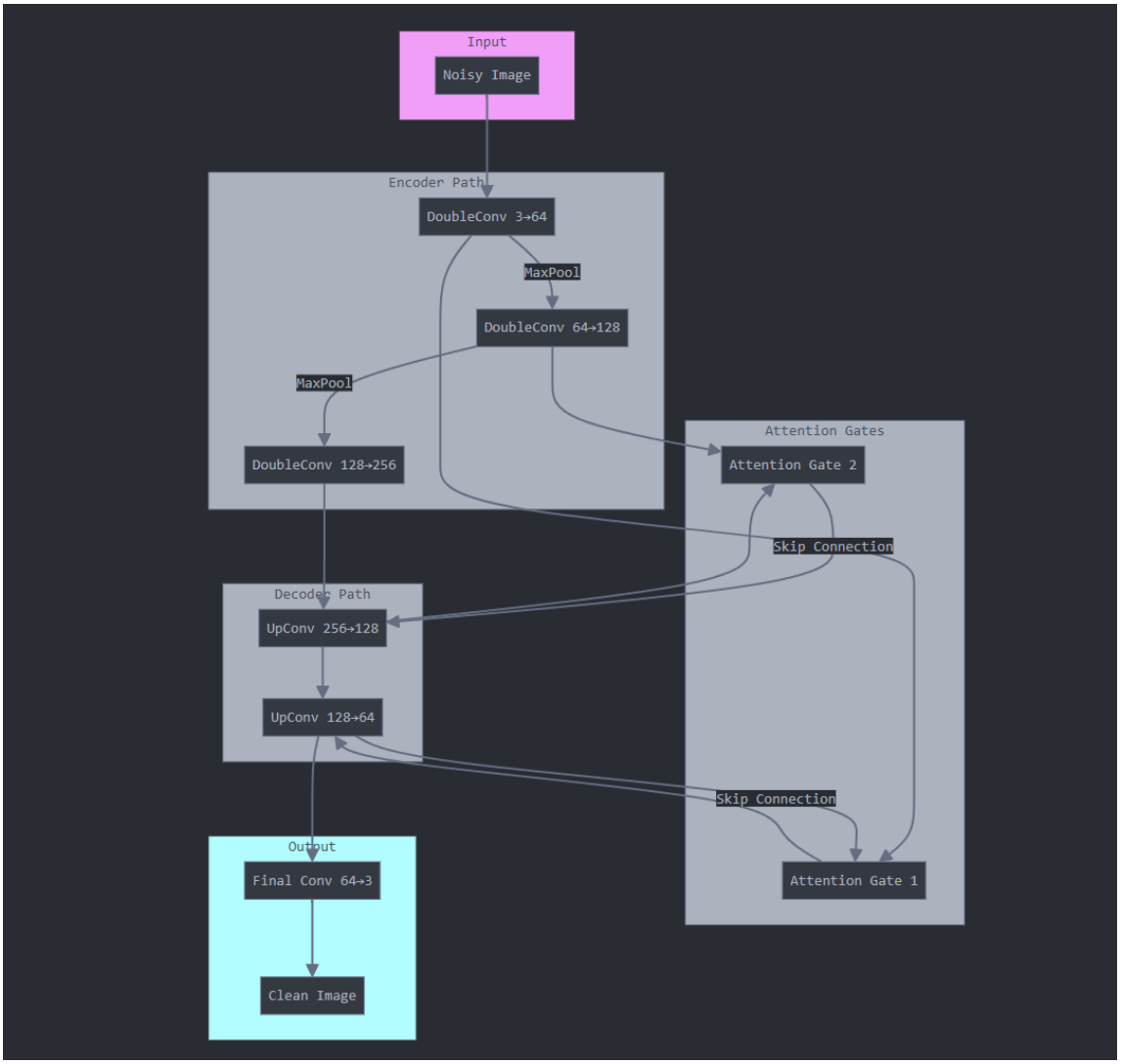
COMPARISON TABLE

Method	SNR Dark	SNR Mid	SNR Light	Edge (Sobel)	Edge (Laplacian)
Original	7.74	9.53	16.08	12.58	9.77
Median	8.44	10.22	18.54	7.77	5.16
Bilateral	8.68	10.90	19.90	6.84	5.71
Gaussian	8.37	10.27	18.17	12.42	10.35
FFDNet	8.19	9.81	16.16	7.22	1.86

Custom Model Components Diagram –

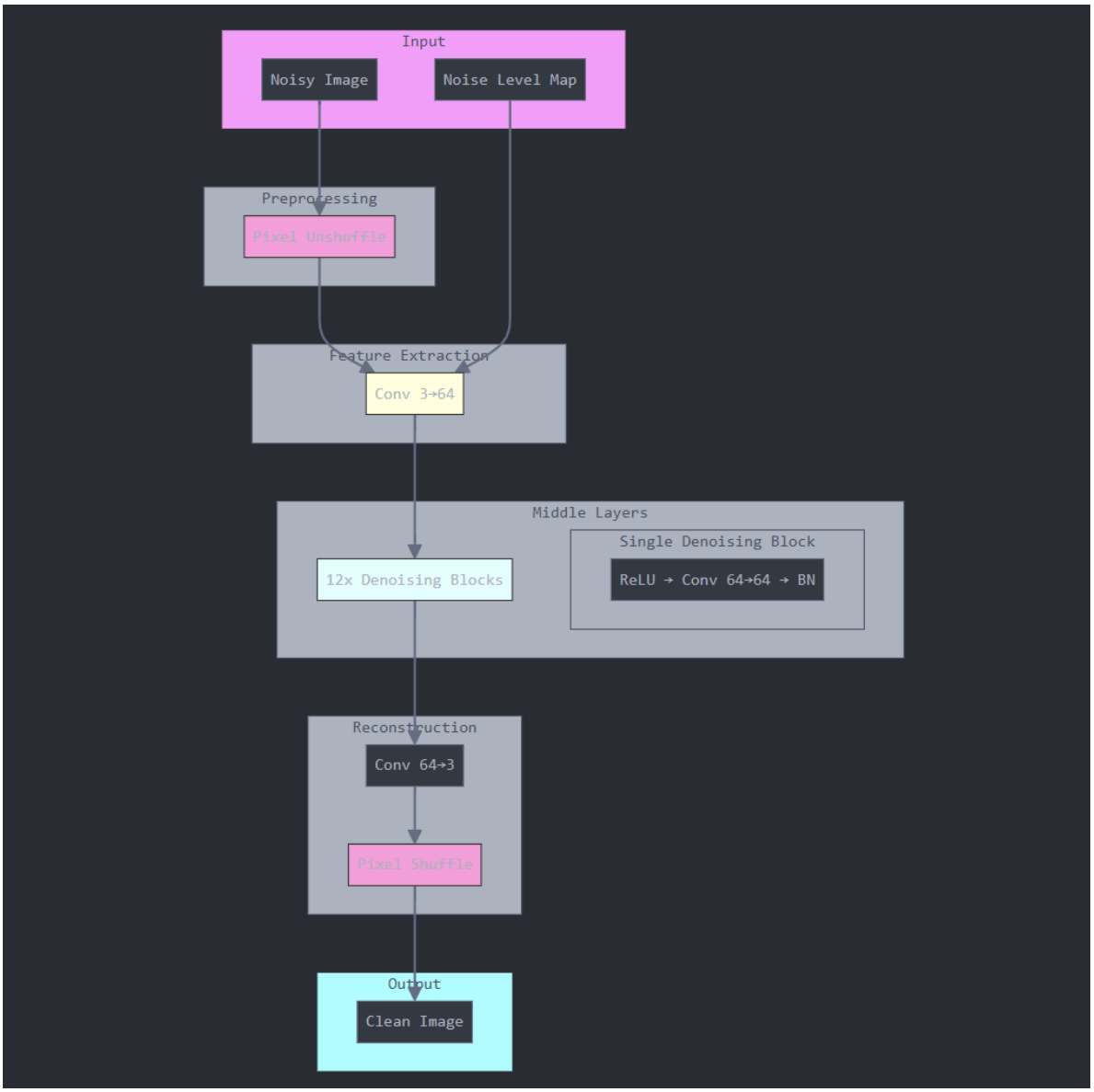
(Though we didn't use it on outputs because it still needed some training

And the results were poor)



- Input: Takes a noisy image as input
- Encoder Path: Three DoubleConv blocks with max pooling between them
    - Progressively increases channels (3→64→128→256)
- Decoder Path: Two upconvolution blocks
    - Progressively decreases channels back (256→128→64)
- Attention Gates: Two attention mechanisms that:
    - Process skip connections from encoder to decoder
    - Help focus on relevant features
- Skip Connections: Connect encoder features to decoder through attention gates
- Output: Final convolution layer that produces the clean image

FFDNET Component diagram –



1. Input:
    - Takes a noisy image
    - Takes a noise level map (unique to FFDNet)
2. Preprocessing:
    - Pixel Unshuffle: Downscales image by rearranging pixels
3. Feature Extraction:
    - Initial convolution that combines image features with noise level information
4. Middle Layers:
    - 12 Denoising blocks
    - Each block contains: ReLU → Convolution → Batch Normalization
5. Reconstruction:
    - Final convolution
    - Pixel Shuffle to restore original resolution

Key differences from the previous model:

- No encoder-decoder structure
- No skip connections
- Uses noise level map as input
- Operates on downscaled images for efficiency
- Uses pixel shuffle/unshuffle for resolution changes